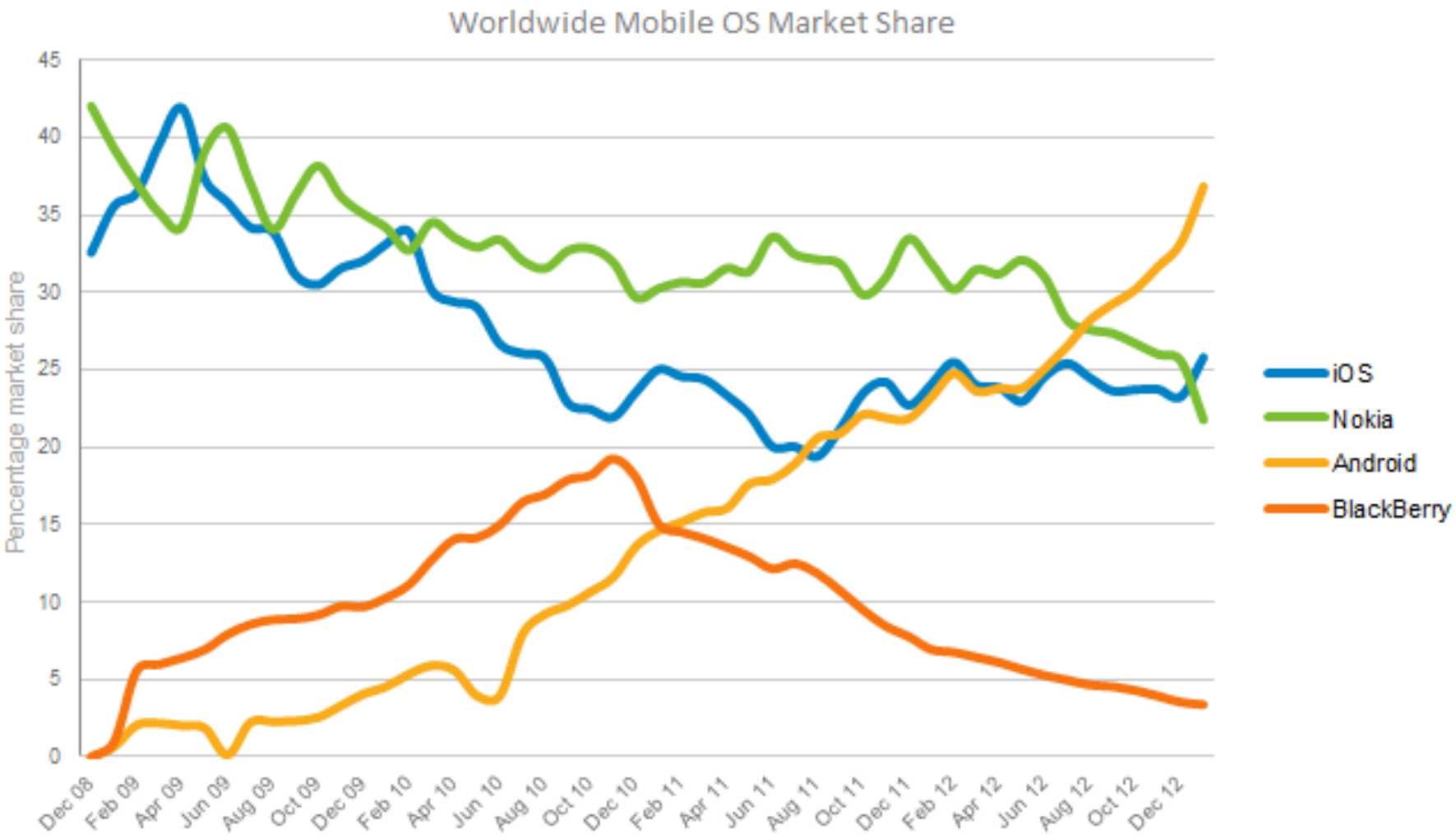
Mobile Platform Security Models

John Mitchell

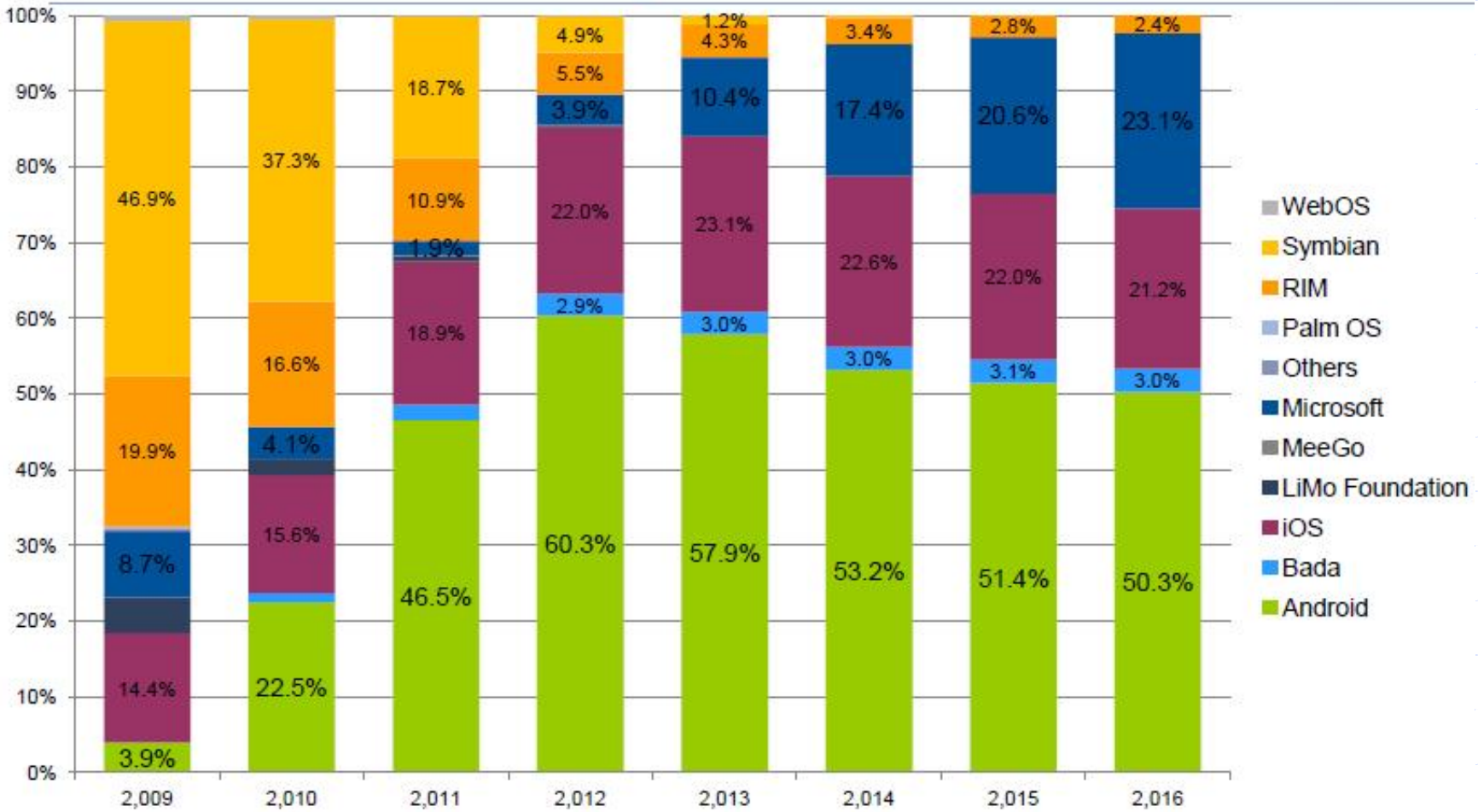
Outline

- Introduction: platforms and attacks
- Apple iOS security model
- Android security model
- Windows 7, 8 Mobile security model

Mobile phone market share



Gartner Forecast Estimates Mobile OS Sales by Market Share (2009-2016)



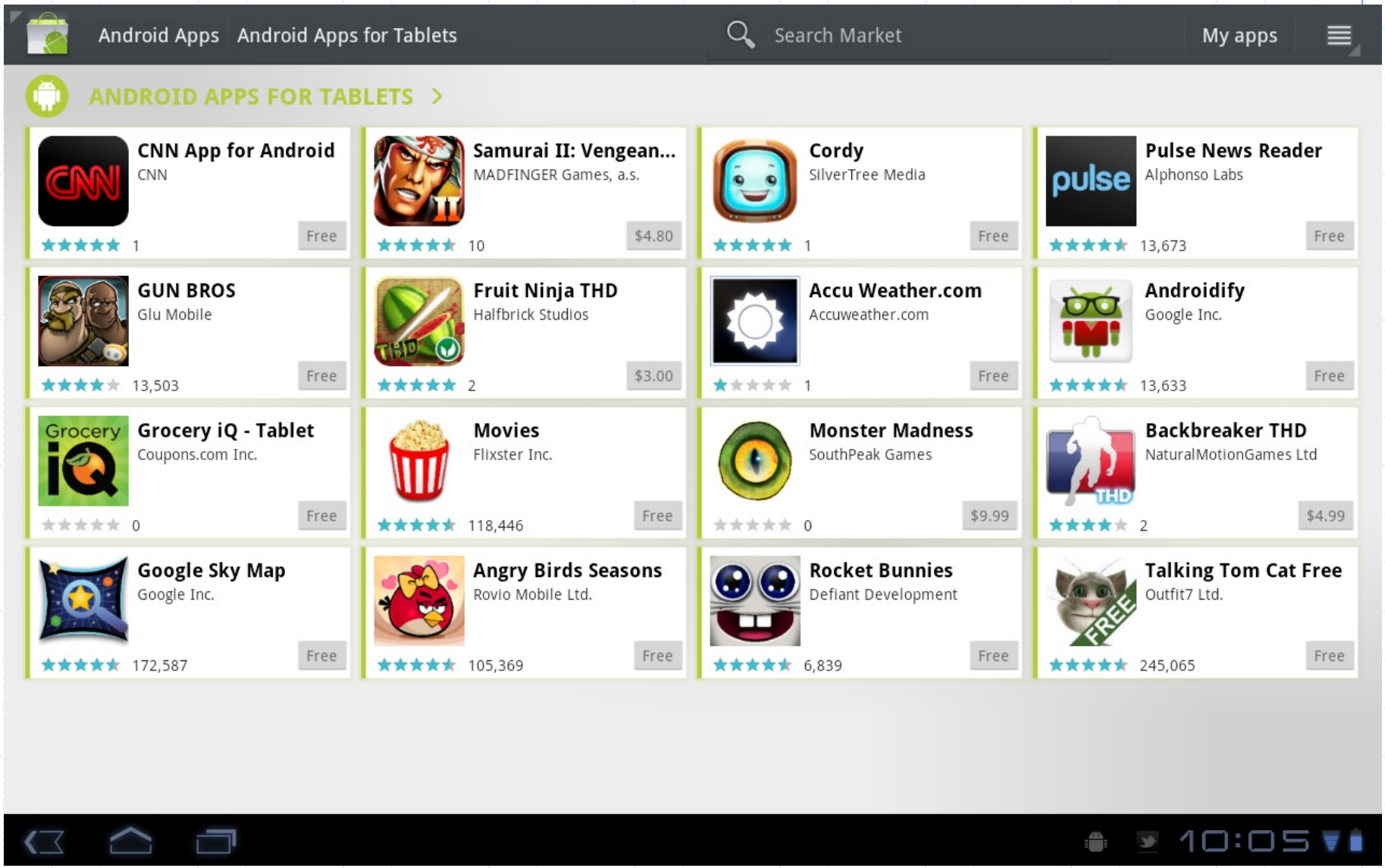
Source: Gartner

Forecast: Mobile Devices by Open Operating System, Worldwide, 2009-2016, 2Q12 Update

9

Gartner

Many mobile apps



Business

iPhone is ready for work. Manage projects, track stocks, monitor finances, and more with these 9–5 apps.

View business apps in the App Store ▸

Education

Keep up with your studies using intelligent education apps like King of Math and Stack the States.

View education apps in the App Store ▸

Entertainment

Kick back and enjoy the show. Or find countless other ways to entertain yourself. These apps offer hours of viewing pleasure.

View entertainment apps in the App Store ▸

Family & Kids

Turn every night into family night with interactive apps that are fun for the whole house.

View family and kids apps in the App Store ▸

Finance

Create budgets, pay bills, and more with financial apps that take everything into account.

View finance apps in the App Store ▸

Food & Drink

Hungry? Thirsty? A little of both? Learn new recipes, drinks, and the secrets behind what makes a great meal.

View food and drink apps in the App Store ▸

Games

The world's best phone is also the world's best portable gaming device. From first-person shooters to strategy, iPhone knows how to have a good time.

View games in the App Store ▸

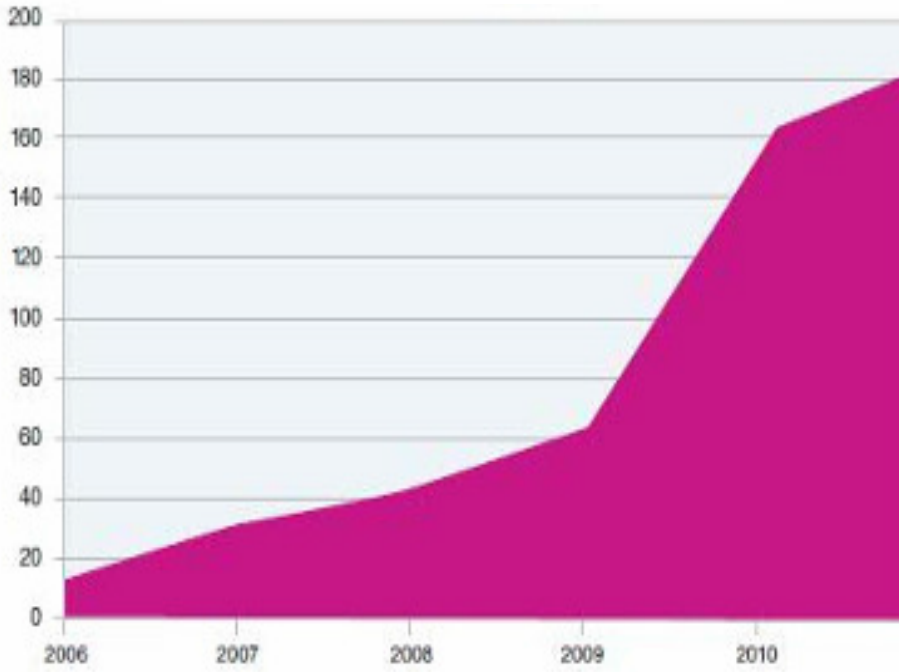
Health & Fitness

When you need a little motivation, let apps like Nike+ FuelBand, Couch-to-5K, and more help you drop pounds and tone muscles.

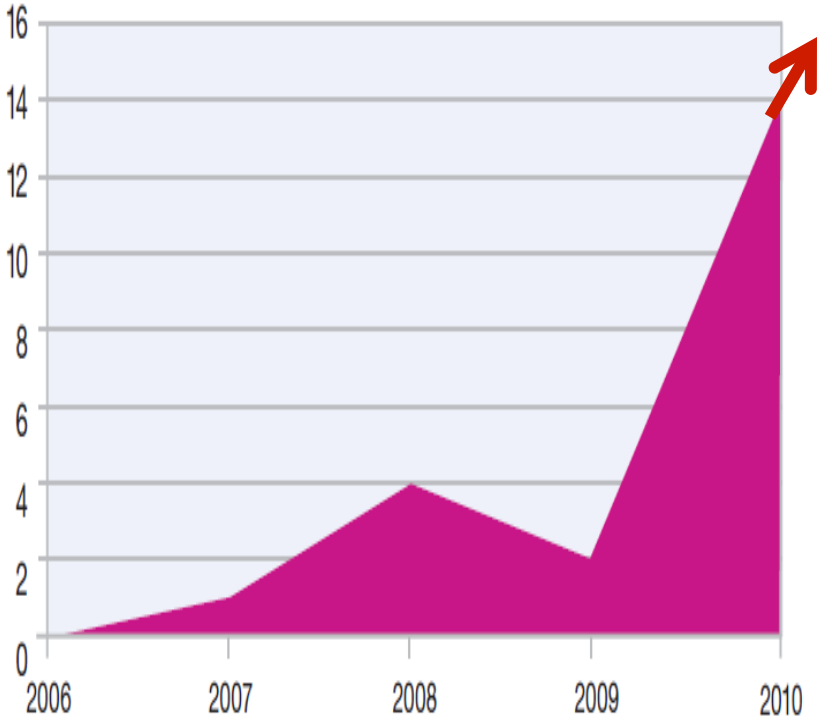
View health and fitness apps in the App Store ▸

Mobile Operating Systems

Mobile OS Vulnerabilities



Mobile OS Exploits



Source: IBM X-Force, Mar 2011

Two attack vectors

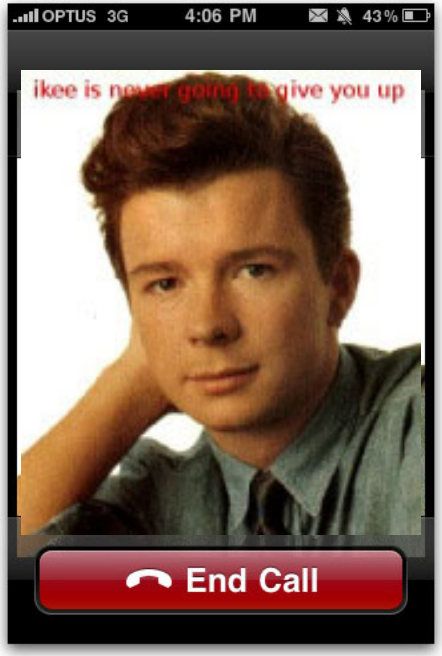
- Web browser
- Installed apps

Both increasing in prevalence and sophistication

Mobile malware attacks

- Unique to phones:
  - Premium SMS messages
  - Identify location
  - Record phone calls
  - Log SMS
- Similar to desktop/PCs:
  - Connects to botmasters
  - Steal data
  - Phishing
  - Malvertising

Mobile malware examples



- DroidDream (Android)
  - Over 58 apps uploaded to Google app market
  - Conducts data theft; send credentials to attacke
- Ikee (iOS)
  - Worm capabilities (targeted default ssh pwd)
  - Worked only on jailbroken phones with ssh installed (could have been worse)
- Zitmo (Symbian,BlackBerry,Windows,Android)
  - Propagates via SMS; claims to install a "security certificate"
  - Captures info from SMS; aimed at defeating 2-factor auth
  - Works with Zeus botnet; timed with user PC infection

Comparison between platforms

- Operating system (recall security features from lecture 5)
  - Unix
  - Windows
- Approval process for applications
  - Market: Vendor controlled/Open
  - App signing: Vendor-issued/self-signed
  - User approval of permission
- Programming language for applications
  - Managed execution: Java, .Net
  - Native execution: Objective C

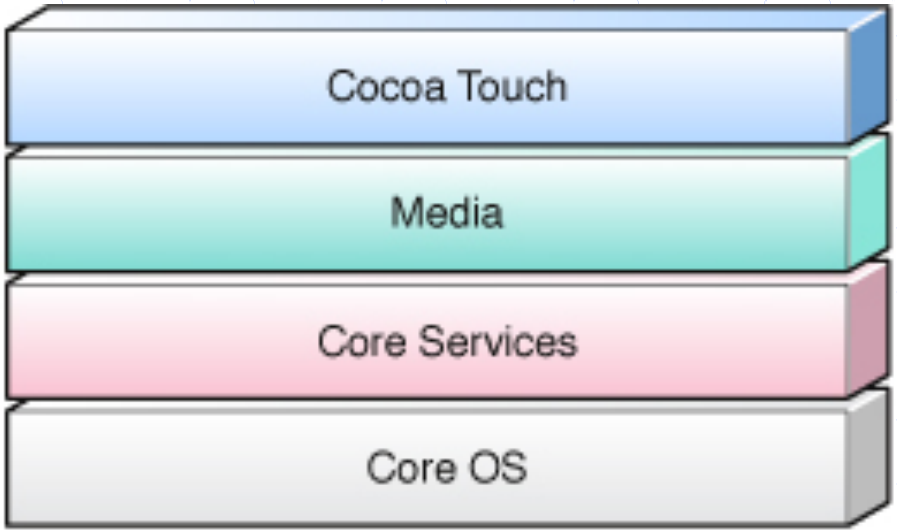
Outline

- Introduction: platforms and attacks
- Apple iOS security model
- Android security model
- Windows 7 Mobile security model

Apple iOS

From: iOS App Programming Guide

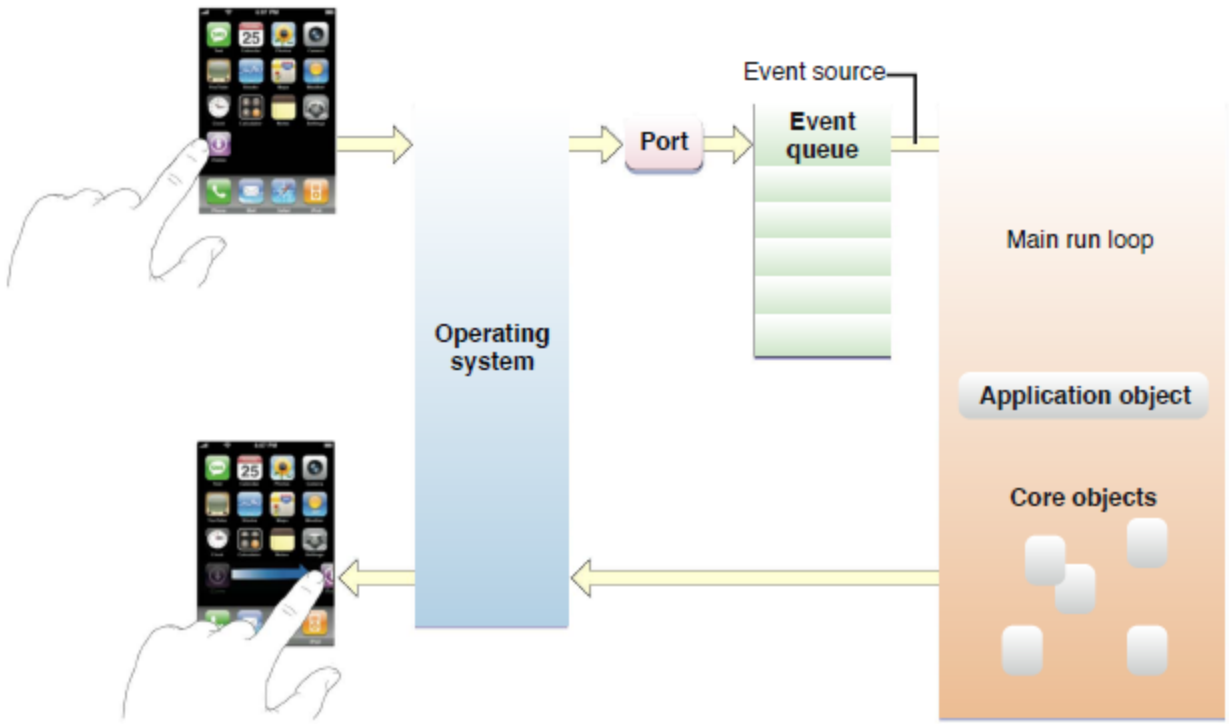
iOS Platform



- Kernel: based on Mach kernel like Mac OS X
- Core OS and Core Services: APIs for files, network, … includes SQLite, POSIX threads, UNIX sockets
- Media layer: supports 2D and 3D drawing, audio, video
- Cocoa Touch: Foundation framework, OO support for collections, file management, network operations; UIKit

Implemented in C and Objective-C

iOS Application Development



- Apps developed in Objective-C using Apple SDK
- Event-handling model based on touch events
- Foundation and UIKit frameworks provide the key services used by all iOS applications

Apple iOS Security

- Device security
  - Prevent unauthorized use of the device
- Data security
  - Protect data at rest; device may be lost or stolen
- Network security
  - Networking protocols and encryption of data in transmission
- App security
  - Secure platform foundation

Reference: http://images.apple.com/iphone/business/docs/iOS_Security.pdf

Device Security: passcodes

- Strong passcodes
- Passcode expiration
- Passcode reuse history
- Maximum failed attempts
- Over-the-air passcode enforcement
- Progressive passcode timeout

Data Security

- Hardware encryption
- Remote wipe
- Local wipe
- Encrypted Configuration Profiles
- Encrypted iTunes backups

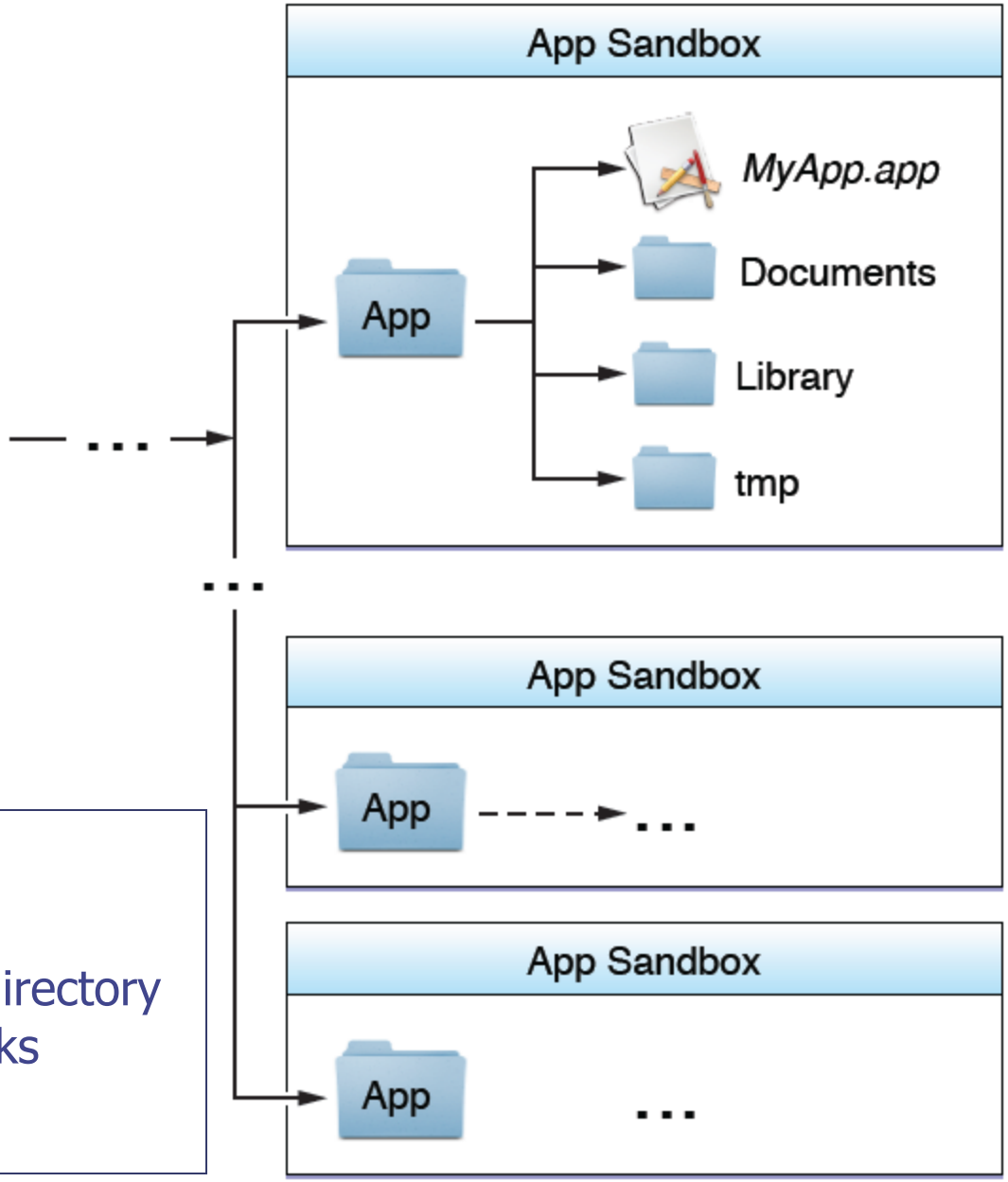
Network Security

- Current accepted network security protocols
  - IPSec, L2TP, PPTP VPN
  - SSL VPN via App Store apps
  - SSL/TLS with X.509 certificates
  - WPA/WPA2 Enterprise with 802.1X

App Security

- Runtime protection
  - System resources, kernel shielded from user apps
  - App "sandbox" prevents access to other app's data
  - Inter-app communication only through iOS APIs
  - Code generation prevented
- Mandatory code signing
  - All apps must be signed using an Apple-issued certificate
- Application data protection
  - Apps can take advantage of built-in hardware encryption

iOS Sandbox



- Limit app's access to files, preferences, network, other resources
- Each app has own sandbox directory
- Limits consequences of attacks
- Same privileges for each app

Comparison

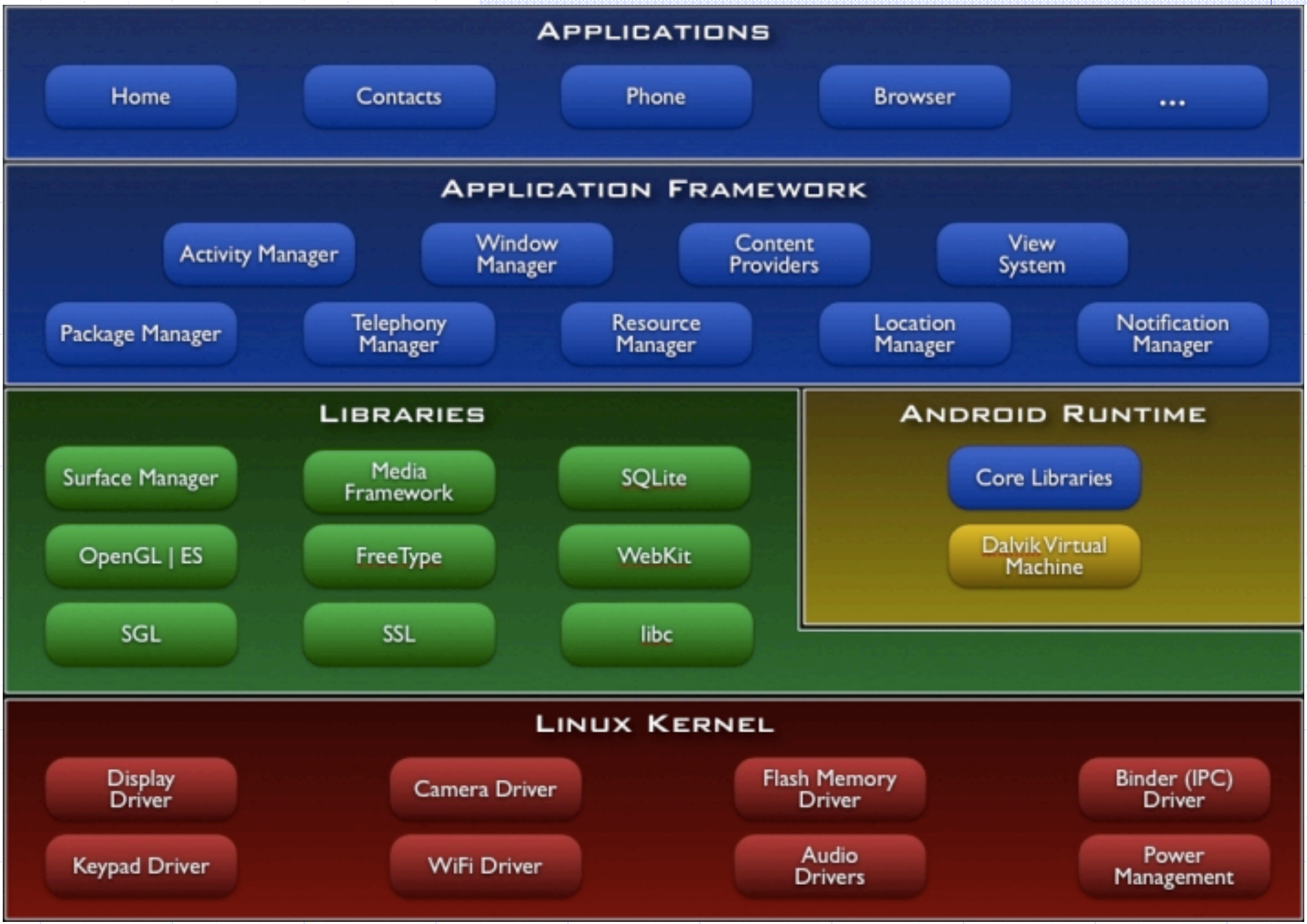
	iOS	Android	Windows
Unix	x		
Windows			
Open market			
Closed market	x		
Vendor signed	x		
Self-signed			
User approval of permissions			
Managed code			
Native code	x		

Outline

- Introduction: platforms and attacks
- Apple iOS security model
- Android security model
- Windows 7, 8 Mobile security model

Android

- Platform outline:
  - Linux kernel, browser, SQL-lite database
  - Software for secure network communication
    - Open SSL, Bouncy Castle crypto API and Java library
  - C language infrastructure
  - Java platform for running applications
  - Also: video stuff, Bluetooth, vibrate phone, etc.


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

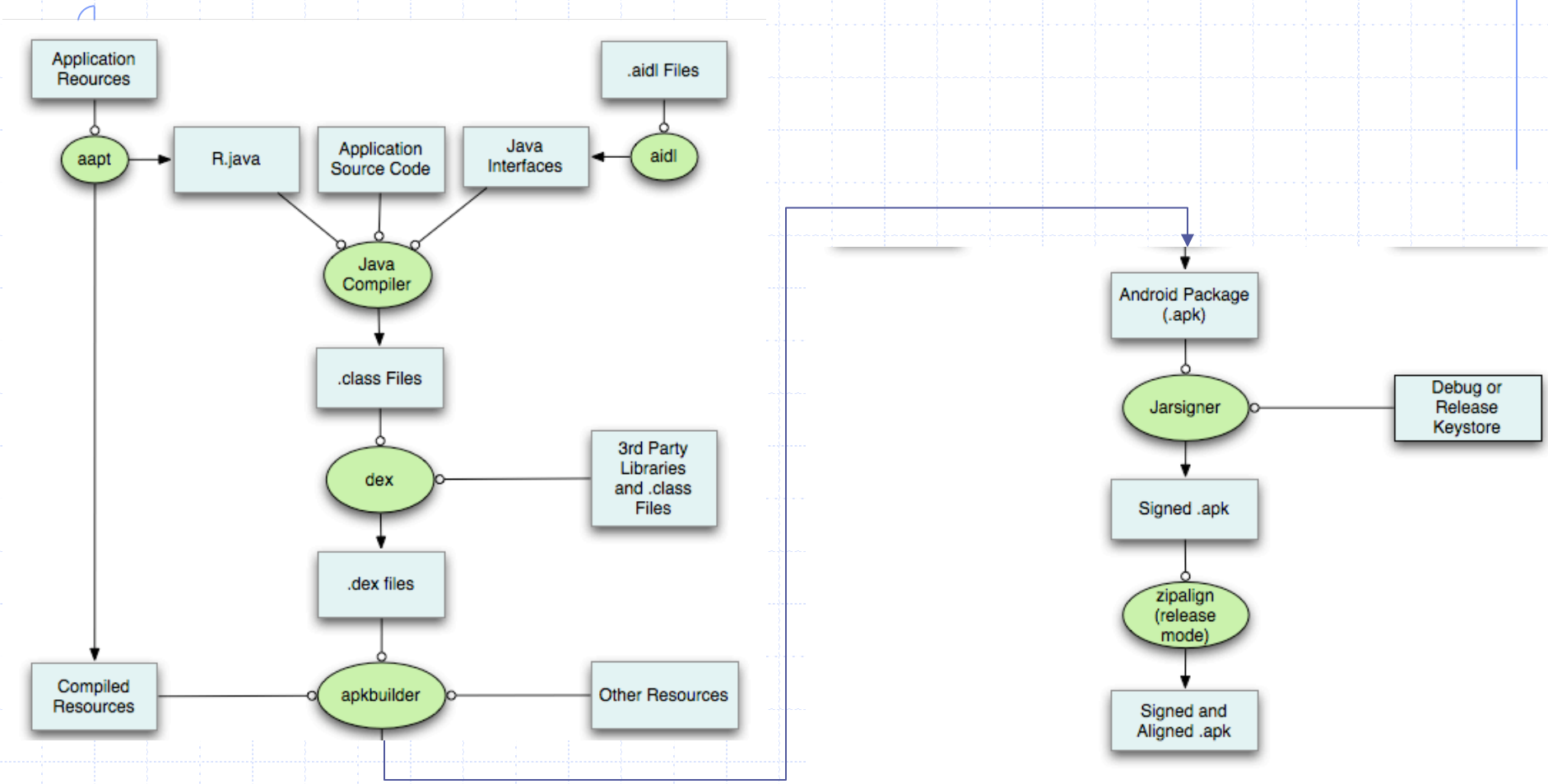
Android market

- Self-signed apps
- Permissions granted on user installation
- Open
  - Bad applications may show up on market
  - Shifts focus from remote exploit to privilege escalation

Security Features

- Isolation
  - Multi-user Linux operating system
  - Each application normally runs as a different user
- Communication between applications
  - May share same Linux user ID
    - Access files from each other
    - May share same Linux process and Dalvik VM
  - Communicate through application framework
    - "Intents," based on Binder, discussed in a few slides
- Battery life
  - Developers must conserve power
  - Applications store state so they can be stopped (to save power) and restarted – helps with DoS

Application development process



Application development concepts

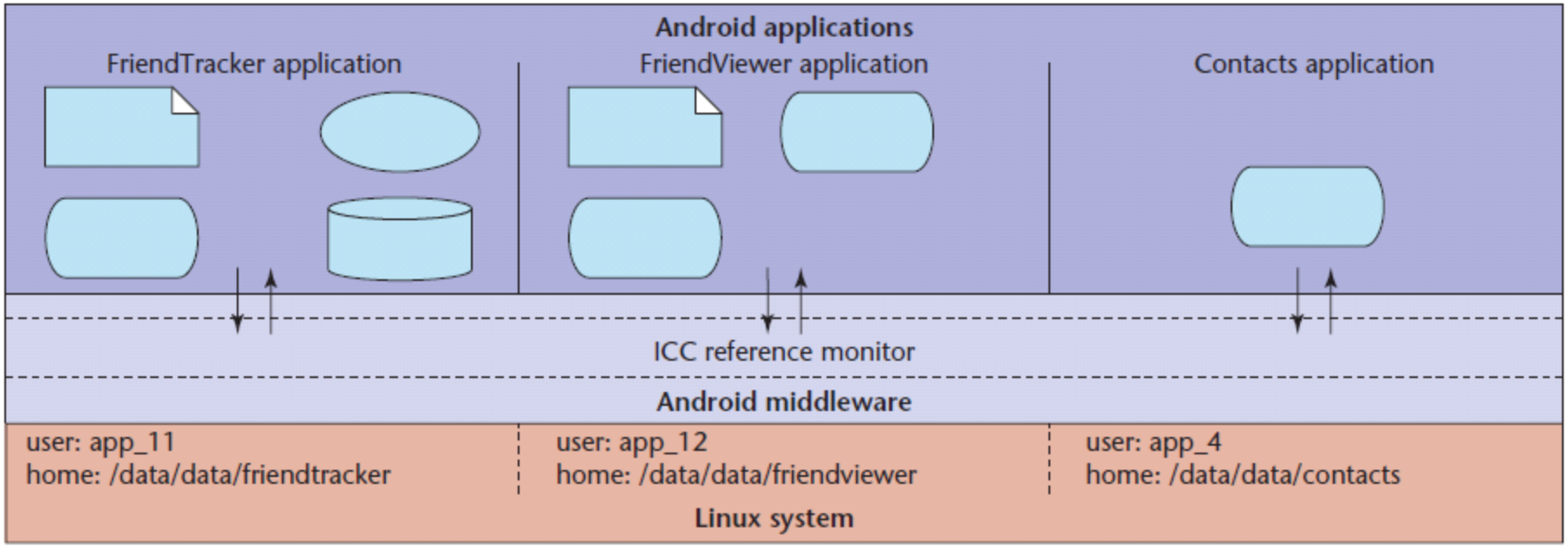
- Activity – one-user task
  - Example: scroll through your inbox
  - Email client comprises many activities
- Service – Java daemon that runs in background
  - Example: application that streams an mp3 in background
- Intents – asynchronous messaging system
  - Fire an intent to switch from one activity to another
  - Example: email app has inbox, compose activity, viewer activity
    - User click on inbox entry fires an intent to the viewer activity, which then allows user to view that email
- Content provider
  - Store and share data using a relational database interface
- Broadcast receiver
  - "mailboxes" for messages from other applications

Exploit prevention

- 100 libraries + 500 million lines new code
  - Open source -> public review, no obscurity
- Goals
  - Prevent remote attacks, privilege escalation
  - Secure drivers, media codecs, new and custom features
- Overflow prevention
  - ProPolice stack protection
    - First on the ARM architecture
  - Some heap overflow protections
    - Chunk consolidation in DL malloc (from OpenBSD)
- ASLR
  - Avoided in initial release
    - Many pre-linked images for performance
  - Developed and contributed by Bojinov, Boneh

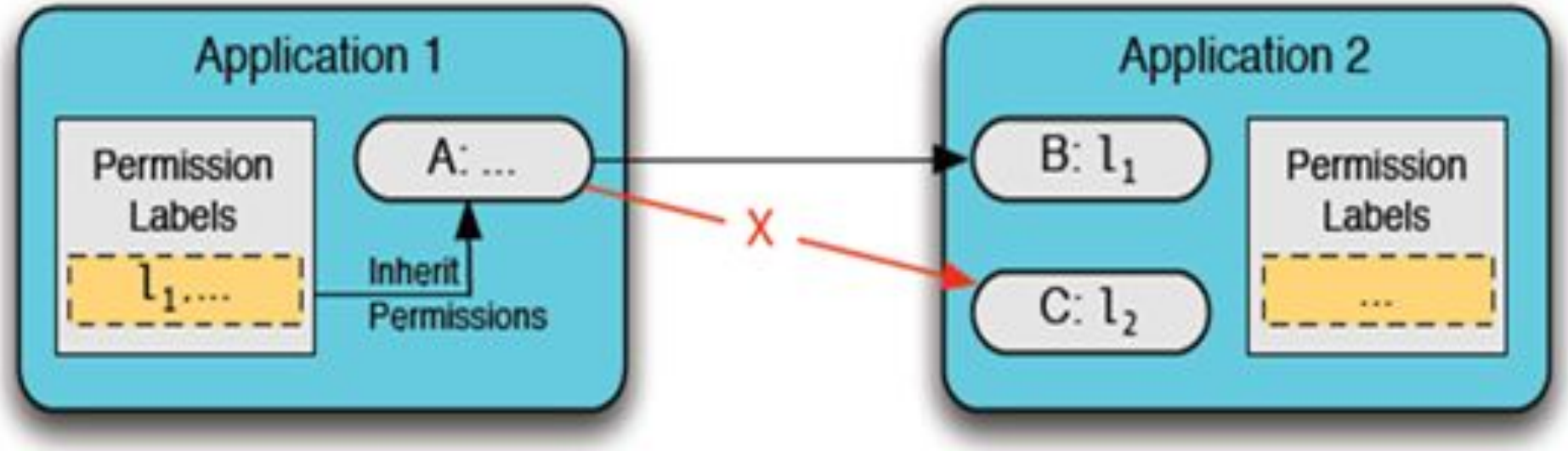
Application sandbox

- Application sandbox
  - Each application runs with its UID in its own Dalvik virtual machine
    - Provides CPU protection, memory protection
    - Authenticated communication protection using Unix domain sockets
    - Only ping, zygote (spawn another process) run as root
  - Applications announces permission requirement
    - Create a whitelist model – user grants access
      - But don't want to ask user often – all questions asked as install time
    - Inter-component communication reference monitor checks permissions



- Layers of security
  - Each application executes as its own user identity
  - Android middleware has reference monitor that mediates the establishment of inter-component communication (ICC)

Source: Penn State group Android security paper



MAC Policy Enforcement in Android. This is how applications access components of other applications via the reference monitor. Component A can access components B and C if permission labels of application 1 are equal or dominate labels of application 2.

Source: Penn State group, Android security tutorial

dlmalloc (Doug Lea)

- Stores meta data in band
- Heap consolidation attack
  - Heap overflow can overwrite pointers to previous and next unconsolidated chunks
  - Overwriting these pointers allows remote code execution
- Change to improve security
  - Check integrity of forward and backward pointers
    - Simply check that back-forward-back = back, f-b-f=f
  - Increases the difficulty of heap overflow

Java Sandbox

- Four complementary mechanisms
  - Class loader
    - Separate namespaces for separate class loaders
    - Associates *protection domain* with each class
  - Verifier and JVM run-time tests
    - NO unchecked casts or other type errors, NO array overflow
    - Preserves private, protected visibility levels
  - Security Manager
    - Called by library functions to decide if request is allowed
    - Uses protection domain associated with code, user policy

Comparison: iOS vs Android

- App approval process
  - Android apps from open app store
  - iOS vendor-controlled store of vetted apps
- Application permissions
  - Android permission based on install-time manifest
  - All iOS apps have same set of "sandbox" privileges
- App programming language
  - Android apps written in Java; no buffer overflow…
  - iOS apps written in Objective-C

See also: http://palisade.plynt.com/issues/2011Oct/android-vs-ios/

Comparison

	iOS	Android	Windows
Unix	x	x	
Windows			
Open market		x	
Closed market	x		
Vendor signed	x		
Self-signed		x	
User approval of permissions		x	
Managed code		x	
Native code	x		

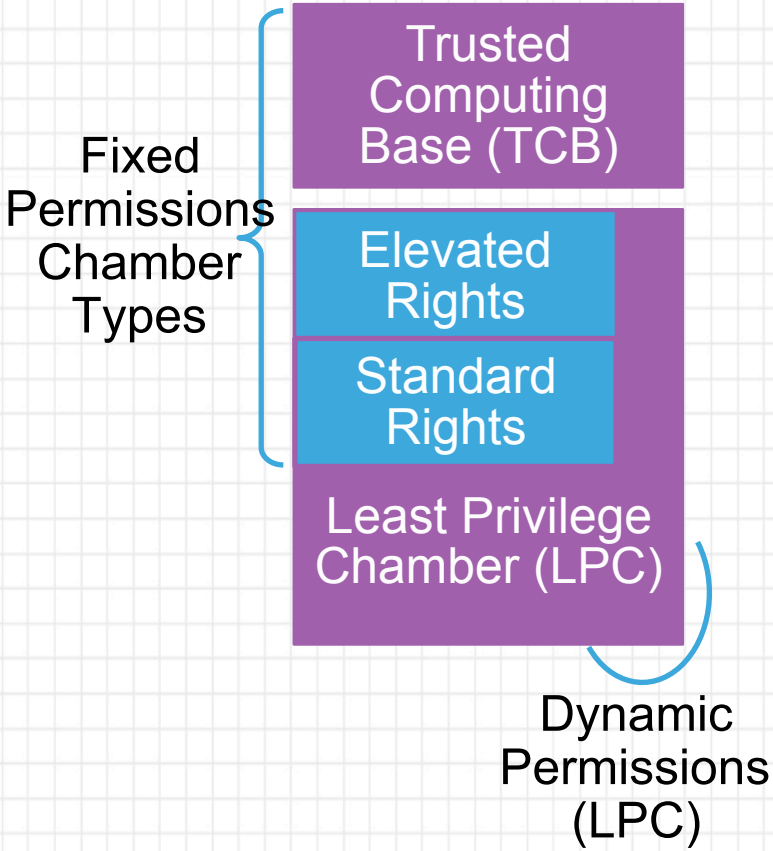
Outline

- Introduction: platforms and attacks
- Apple iOS security model
- Android security model
- Windows Phone 7, 8 security model

Windows Phone 7, 8 security

- Secure boot
- All binaries are signed
- Device encryption
- Security model with isolation, capabilities

Windows Phone 7 security model



Policy system

Central repository of rules
3-tuple {Principal, Right, Resource}

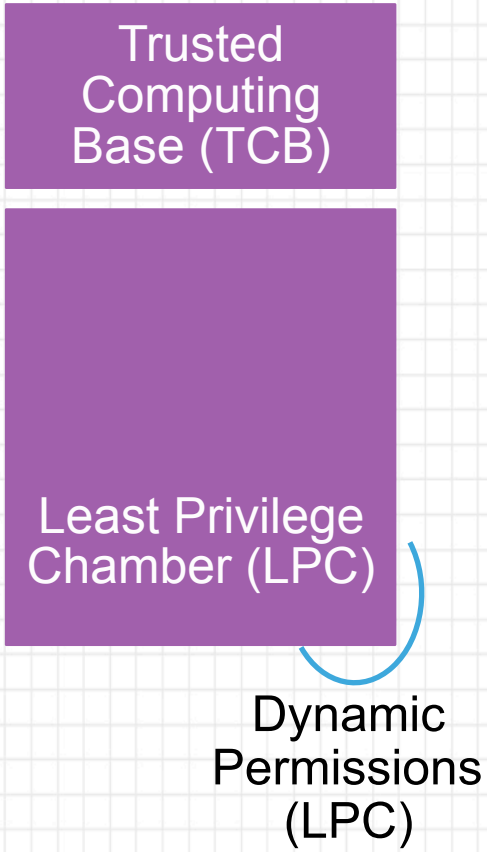
Chamber Model

Chamber boundary is security boundary
Chambers defined using policy rules
4 chamber types, 3 fixed size, one can be expanded with capabilities (LPC)

Capabilities

Expressed in application manifest
Disclosed on Marketplace
Defines app's security boundary on phone

Windows Phone 8 security model



Similar to WP7

WP8 chambers are built on the Windows security infrastructure

Services and Application all in chambers

WP8 has a richer capabilities list

Windows Phone OS 7.0 security model

- Principles of isolation and least privilege
- Each chamber
  - Provides a security and isolation boundary
  - Is defined and implemented using a policy system
- The security policy of a chamber
  - Specifies the OS capabilities that processes in that chamber can access

Isolation

- Every application runs in own isolated chamber
  - All apps have basic permissions, incl a storage file
  - Cannot access memory or data of other applications, including the keyboard cache.
- No communication channels between applications, except through the cloud
- Non-MS applications distributed via marketplace stopped in background
  - When user switches apps, previous app is shut down
  - Reason: application cannot use critical resources or communicate with Internet–based services while the user is not using the application

Four chamber types

- Three types have fixed permission sets
- Fourth chamber type is capabilities-driven
  - Applications that are designated to run in the fourth chamber type have capability requirements that are honored at installation and at run-time

Overview of four chambers

- Trusted Computing Base (TCB) chamber
  - unrestricted access to most resources
  - can modify policy and enforce the security model.
  - kernel and kernel-mode drivers run in the TCB
  - Minimizing the amount of software that runs in the TCB is essential for minimizing the Windows Phone 7, 8 attack surface

Overview of four chambers

- Elevated Rights Chamber (ERC)
  - Can access all resources except security policy
  - Intended for services and user-mode drivers
- Standard Rights Chamber (SRC)
  - Default for pre-installed applications that do not provide device-wide services
  - Outlook Mobile is an example that runs in the SRC
- Least Privileged Chamber (LPC)
  - Default chamber for all non-Microsoft applications
  - LPCs configured using capabilities (see next slide)

Granting privileges to applications

- Goal: Least Privilege
  - Application gets capabilities needed to perform all its use cases, but no more
- Developers
  - Use the capability detection tool to create the capability list
  - The capability list is included in the application manifest
- Each application discloses its capabilities to the user,
  - Listed on Windows Phone Marketplace.
  - Explicit prompt upon application purchase
  - Disclosure within the application, when the user is about to use the location capability for the first time.

Windows Phone 7 “Capabilities”

- W7 Capability: a resource associated with user privacy, security, cost, or business concerns
- Examples: geographical location information, camera, microphone, networking, and sensors.

Managed code

- Application development model uses of managed code only

.NET Code Access Security

- Default Security Policy is part of the .NET Framework
  - Default permission for code access to protected resources
- Permissions can limit access to system resources.
  - Use EnvironmentPermission class for environment variables access permission.
  - The constructor defines the level of permission (read, write, …)
- Deny and Revert
  - The Deny method of the permission class denies access to the associated resource
  - The RevertDeny method will cause the effects of any previous Deny to be cancelled

Example: code requires permission

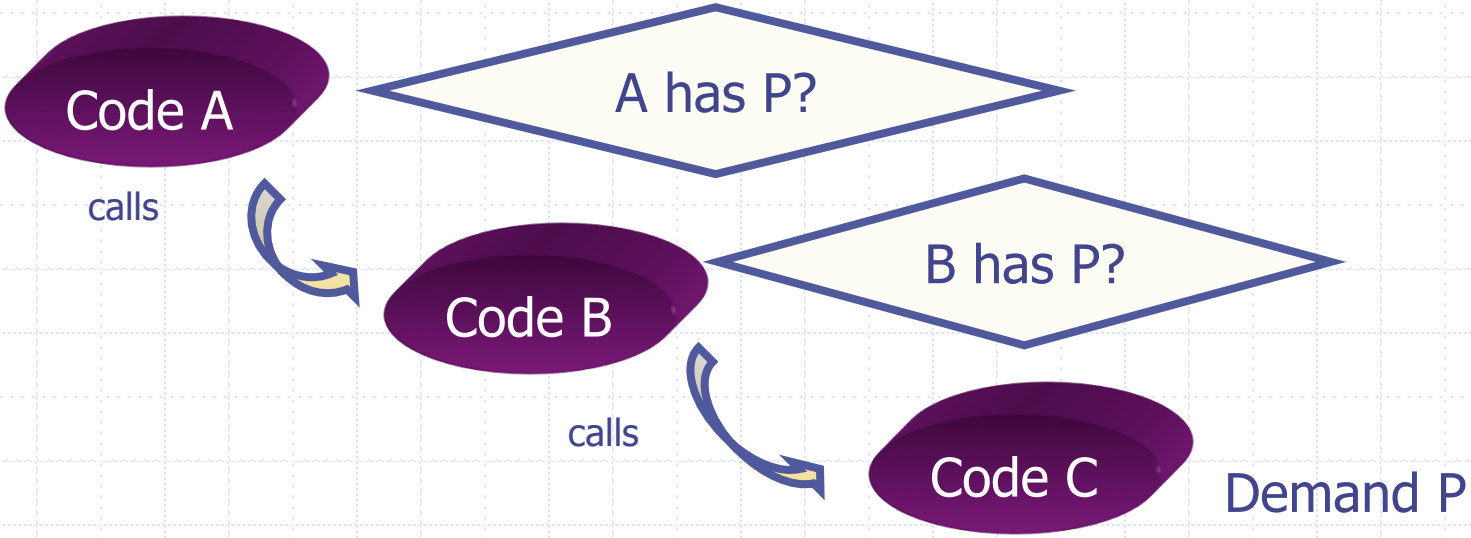
```
class NativeMethods
{
    // This is a call to unmanaged code. Executing this method
    // requires the UnmanagedCode security permission. Without
    // this permission, an attempt to call this method will throw a
    // SecurityException:
    [DllImport("msvcrt.dll")]
    public static extern int puts(string str);
    [DllImport("msvcrt.dll")]
    internal static extern int _flushall();
}
```

Example: Code denies permission not needed

```
[SecurityPermission(SecurityAction.Deny, Flags =
    SecurityPermissionFlag.UnmanagedCode)]
  private static void MethodToDoSomething()
  {   try
      {
          Console.WriteLine(" … ");
          SomeOtherClass.method();
       }
      catch (SecurityException)
      {
          …
      }
  }
```

.NET Stackwalk

- Demand must be satisfied by all callers
  - Ensures all code in causal chain is authorized
  - Cannot exploit other code with more privilege



Stackwalk: Assert

- The Assert method can be used to limit the scope of the stack walk
  - Processing overhead decreased
  - May inadvertently result in weakened security

Comparison between platforms

- Operating system
  - Unix
  - Windows
- Approval process for applications
  - Market: Vendor controlled/Open
  - App signing: Vendor-issued/self-signed
  - User approval of permissions
- Programming language for applications
  - Managed execution: Java, .Net
  - Native execution: Objective C

Comparison

	iOS	Android	Windows
Unix	x	x	
Windows			x
Open market		x	
Closed market	x		x
Vendor signed	x		
Self-signed		x	x
User approval of permissions		x	7-> 8
Managed code		x	x
Native code	x		

Conclusion

- Introduction: platforms and attacks
- Apple iOS security model
- Android security model
- Windows 7, 8 Mobile security model